

GIGABIT ETHERNET BRIDGE WITH PARALLEL FIFO STRUCTURE**Field of the invention**

- 5 The present invention relates to a network bridge for transforming a network data stream from a higher bandwidth data stream to a lower bandwidth data stream.

Background

- 10 The Synchronous Optical Network (SONET) includes a set of signal rate multiples for transmitting digital signals over optical fiber. The base rate (OC1) is 51.84 Mb/s. Certain multiples of the base rate are defined as standard data rates. Ethernet, asynchronous transfer mode and other protocols make use of some of the optical carrier levels. Transforming between optical carrier levels in SONET is a critical issue for the effective
15 use of network storage systems, Ethernet routers, Ethernet bridges and switchers. In particular, transforming from 10-Gb/s data streams (SONET OC192) to 2.5-Gb/s data streams (SONET OC48) is particularly desirable for the following reasons:

1. Data can be transformed from an exterior network that has higher bandwidth,
20 traffic, cost to an interior network that has lower bandwidth, traffic and cost.
2. For the NIC (network interface card), the transformation provides a connection between the physical layer framer chip at 10 Gb/sec and the physical layer framer chip at 2.5 Gb/sec. In particular, the transformation is used for connection
25 between advanced chips such as, Wire-speed Ganges 10 Gigabit framer, Yukon 2.5-Gb/s framer and Forte wire-speed edge router IC. Details of these chips are given below.
- Ganges II (Part Number: S19202CBI20, Manufacturer: Applied Micro Circuits
30 Corporation) is a highly integrated VLSI device that provides full duplex mapping of Packets or ATM cells into SONET/SDH payloads at rates up to 9.95 Gb/s.

- Yukon (Part Number: S4803TBI11, Manufacturer: Applied Micro Circuits Corporation) implements SONET/SDH processing and full-duplex ATM/POS mapping functions for STS-48/STM-16 data streams at a rate of 2.5Gb/s.
- 5 • Forte (Part Number: ENT3048, Manufacturer: Entridia) for IP/MPLS based edge routers that features two full duplex 2.5-Gbp/s data ports that support either the FlexBus-3 standard or the POS-PHY level 3, 32 bit system interface standard.

Relevant differences between SONET 2.5-Gb/s interfaces and 10-Gb/s interfaces are
10 summarized as follows:

- different transmitting frequencies: Transmitting frequencies for SONET OC48, which is supported by POS3 and Flexbus3, are from 77 MHz to 100 MHz. Transmitting frequencies for SONET OC192, which is supported by POS4,
15 Flexbus4, are from 155 MHz to 200 MHz.
- different transmitting data widths: The data width supported by SONET OC48 is 32 bits and, the data width supported by SONET OC192 is 64 bits.
- 20 • different parity generating methods: For SONET OC48, the parity signal is generated from its 32 bits transmitting data width. For SONET OC192, four separate parity signals are independently generated from its 16 bits transmitting data width.

25 Typically, a parallel first-in-first-out (FIFO) register structure is adopted for network elements transforming OC192 to OC48 data streams. Incoming data from the OC192 interface is fed into two FIFOs in turn. Accordingly, data read from these parallel FIFOs may not be in order and the resulting data stream may be "zigzagged" due to the different depths of data values in the respective FIFOs. That is, different amounts of data are
30 queued in different FIFOs, causing the order of the data values transmitted at the output to be different from the order of the data values received at the input.

A number of existing publications describe solutions for data interface transformation. Each of the United States patent publications listed below relates to different interface standards, and implements different structures and control mechanism.

- 5 1. United States patent no 6,222,848 issued April 24, 2001 to Hayward et al and
entitled: "Gigabit ethernet interface to synchronous optical network (SONET)
ring".
2. United States patent no 5,835,496 issued November 10, 1998 to Yeung et al and
10 entitled: "Method and apparatus for data alignment".
3. United States patent no 5,469,558 issues November 21, 1995 to Lieberman et al
and entitled: "Dynamically reconfigurable memory system with programmable
controller and FIFO buffered data channels".

15

None of the above-listed publications disclose any procedures for transforming from OC192 (Flexbus4, POS4) to OC48 (Flexbus3, POS3) data streams using a parallel FIFO structure.

- 20 In view of the above observations, a need clearly exists for a network bridge having an improved input-output structure for transforming from a faster (that is, higher bandwidth) to a slower (that is, lower bandwidth) data stream.

Summary

25

Implementations of the described techniques are suitable for transforming data packets from a wide (for example, 10-Gb/s) parallel interface (such as SONET, OC192, Flexbus4, Pos4) to a narrow (for example, 2.5-Gb/s) parallel interface (such as SONET, OC48, Flexbus3, Pos3).

30

FIFO structures are used to receive the data from an OC192 data stream, which operates from 155 MHz to 200 MHz and decrease the data speed to half of the incoming data stream. A first negotiation mechanism is described to prevent the data stream "zigzagging" in the space and time domains due to non-alignment of data output from the parallel FIFO structures. The data

stream is transformed into the format of an OC48 data stream, and a second negotiation mechanism is described for transmitting the data stream through the OC48 interface.

In order to eliminate the difference of the FIFO depth while accepting the data stream from OC192 interface, a data inserting mechanism is described in relation to a FIFO write controller. While reading the data out from the FIFOs, the first negotiation mechanism is used to determine in what order data from the FIFOs should be sent. This negotiation mechanism is implemented in relation to a FIFO read controller. Data that was earlier inserted in the incoming data stream is removed from the outgoing data stream.

Description of drawings

Fig. 1 is a schematic representation of the signals that pass between a source OC192 interface, a target OC48 interface and an interposed transformer.

Figs. 2A to 2F are a first subset of schematic representations of the internal functions of the transformer of Fig 1.

Figs. 3A to 3F are a second subset of schematic representations of the internal functions of the transformer of Fig. 1.

Fig. 4 is a third schematic representation of a subset of the internal functions of the transformer of Fig. 1.

Fig. 5 is a schematic representation of the structure of a FIFO used in the source OC192 interface of Fig. 1.

Figs. 6A and 6B are timing diagrams for some of the signals represented in Fig. 1, for different respective data values.

Fig. 7 is a timing diagram some of the signals represented in Figs. 1 to 3 that illustrates data "zigzagging" in the space and time domain.

Fig. 8 is a schematic representation of the structure of a FIFO writer controller used in the transformer of Fig. 1 and represented in Fig. 3A.

Fig. 9 is a timing diagram for signals in the FIFO write controller of Fig. 8.

5

Fig. 10 is a schematic representation of the structure of a FIFO read controller used in the transformer of Fig. 1 and represented in Fig. 3E.

Fig. 11 is a timing diagram for signals in the FIFO read controller of Fig. 10.

10

Fig. 12 is a schematic representation of the structure of a tap generator used in the transformer of Fig. 1 and represented in Fig. 4.

Fig. 13 is a table of pseudo-code of an output controller in the transformer of Fig. 1.

15

Fig. 14 is a timing diagram for signals in the tap generator of Fig. 12.

Fig. 15 is a state transition diagram of a state machine used in the transformer of Fig. 1 and represented in Fig. 12

20

Fig 16 is a schematic representation of transformer 1000, components of which are described with reference to Figs 1 to 15.

Fig. 17 is a flowchart of the process involved in interfacing data streams from the OC192 and OC48 data interfaces of Fig. 1.

25

Fig. 18 is a schematic representation of a network bridge suitable for incorporating techniques described with reference to Figs. 1 to 15.

30

Detailed description

A method, digital hardware, and digital logic are herein described for transforming data streams from a wider data stream to a narrower data stream. This transformation is

described in the context of a transformation interface receives an incoming OC192 data stream that and transmits an outgoing OC48 data stream.

Fig. 1 schematically represents the components involved in converting an incoming OC192 data stream to an outgoing OC48 data stream. An OC192 INTERFACE 2000 operatively interfaces with a TRANSFORMER 1000, which in turn operatively interfaces with an OC48 INTERFACE 3000. The TRANSFORMER 1000 interposed between the OC192 INTERFACE 2000 and the OC48 INTERFACE 3000 is also referred to herein as a TRANSFORMER 1000.

The OC192 INTERFACE 2000 receives an incoming wider data stream at 10Gb/s and is supported by Flexbus4 or Pos4. The OC48 INTERFACE 3000 provides an outgoing narrower data stream at 2.5 Gb/s, and is supported by Flexbus3 or Pos3.

Fig. 1 provides a signal specification of the signals that pass between the OC192 INTERFACE 2000, the TRANSFORMER 1000 and the OC48 INTERFACE 3000. Key signals are described in detail directly below, followed by an overview of other signals with reference to Figs. 2A to 2F.

TX_CLAV/PA 3010 is a FIFO status signal that is generated from the physical layer of the OC48 INTERFACE 3000. This signal 3010 acts as a flow control signal. If this signal 3010 is active, data transmission from the TRANSFORMER 1000 is stopped. Optionally, predetermined data is transmitted to the OC48 INTERFACE 3000 is this signal 3010 is active.

SRX_VALID 2020 is a transmitting valid signal that originates from the OC192 INTERFACE 2000. This signal 2020 represents different transmission modes or states that can indicate that data transmission is either (i) current, (ii) paused or (iii) stopped.

SRX_CHANNEL [3:0] 2030 is a transmitting channel signals that originates from the OC192 INTERFACE 2000. This signal 2030 is used to transmit the OC192 data stream to different destinations. This design can be extended for 4 different OC48 destinations (4 channels) by using this signal. In the present case, only one channel is used as the only destination.

SRX_FRAME 2110 and **SRX_FULL[3:0]** 2120 are both flow control signals that originate from the TRANSFORMER 1000. These signals 2110 and 2120 act as a flow control signal to the OC192 INTERFACE 2000. If these signals 2110 and 2120 are active, then data transmission to the TRANSFORMER 1000 is stopped. Optionally, predetermined data values are transmitted from the OC912 INTERFACE 2000 to the TRANSFORMER 1000.

STX_CLK 2130 is the transmitting clock of OC192 INTERFACE 2000. This clock is the input signal for OC192 INTERFACE 2000. **STX_CLK** 2130 is generated by the TRANSFORMER 2000. In this design, **STX_CLK** 2130 is generated from GLOBAL CLOCK BUFFER 1500.

With reference to Fig. 1, the OC192 INTERFACE 2000 transmits various other signals to the TRANSFORMER 1000. These signals are described with reference to Figs. 2A to 2F, and are listed as follows:

- **SRX_CLK** 2010, a global clock signal
- **SRX_VALID** 2020, valid signal for data.
- **SRX_CHANNEL[3:0]** 2030, signal which carries the channel information
- **SRX_SOCP** 2040, a start of cell/packet signal
- **SRX_DATA_OUT[63:0]** 2050, 64-bit wide incoming data signal
- **SRX_PRTY[3:0]** 2070, a 4-bit wide parity signal for data
- **SRX_PRTY[4]** 2060, special parity signal for controls
- **SRX_EOP** 2080, a end of packet signal
- **SRX_ERROR** 2090, an error signal

- **SRX_SIZE[2:0]** 2100, a 3-bit wide signal which gives the information of valid part of data

5 Also with reference to Fig. 1, the OC192 INTERFACE 2000 transmits various other signals to the TRANSFORMER 1000. These signals are described with reference to Figs. 3F and 4, and are listed as follows:

- **TX_ENB** 3030, a transmission enable signal
- 10 • **TX_EOP** 3040, a transmission end of packet signal
- **TX_SOC/P** 3050, a transmission start of cell/packet signal
- 15 • **TX_ERR** 3060, a transmission error signal
- **TX_PRTY** 3070, a transmission parity signal

20 *Internal structure of the transformer*

Figs. 2A to 2F are a first subset of schematic representations of respective parts of the TRANSFORMER 1000.

25 Fig. 2A schematically represents clock signals in the TRANSFORMER 1000. In Fig. 2A, **SRX_CLK** 2010 is a global clock signal that drives GLOBAL CLOCK BUFFER 1500.

The output clock signal from GLOBAL CLOCK BUFFER 1500 is **srxclk_in** 1502, which operates at 155 MHz to 200 MHz. The **srxclk_in** 1502 signal is input to
30 CLOCK DIVIDER 1510, and outputs a **CLK** 1512 signal at half the frequency (77 MHz to 100 MHz) of the input **srxclk_in** 1502 signal (155 MHz to 200 MHz).

The global clock signal **SRX_CLK** 2010 is divided by CLOCK DIVIDER 1510 to decrease the internal frequency of data transmitting and transforming. Consequently, the

frequency of output clock, CLK 1512 (which operates at 77 MHz to 100 MHz), is only half the frequency of the global clock signal SRX_CLK 2010.

5 STX_CLK 2130 is the input signal for the OC192 INTERFACE 2000 generated by the TRANSFORMER 1000. STX_CLK 2130 is used to get the valid SRX_FRAME 2110 and SRX_FULL[3:0] 2120 signals. STX_CLK 2130 need not synchronized with the STX_CLK 2010.

10 Fig. 2B schematically represents a first level latch IOB/LATCH1 1610. Fig. 2C schematically represents another first level latch IOB/LATCH2 1620. In the first level latch IOB/LATCH1 1610 and IOB/LATCH2 1620, the data signals are triggered using the rising edge of input clock signal srxclock_in 1502 from the GLOBAL CLOCK BUFFER 1500.

15 In Fig. 2B, control signals are input to the first level latch IOB/LATCH1 1610. These control signals are: SRX_CHANNEL [3:0] 2030, SRX_VALID 2020, SRX_SOCP 2040, SRX_EOP 2080, SRX_ERROR 2090, SRX_PRTY [3:0] 2070, SRX_SIZE [2:0] 2100, SRX_PRTY [4] 2060. In Fig. 2C, the incoming 64-bit data signal, SRX DATA_OUT [63:0] 2050 is input to the first level latch IOB/LATCH2 1620.

20

The first level latch IOB/LATCH1 1610 outputs a A_LATCH1 1612 signal, and the first level latch signal IOB/LATCH2 1620 outputs a B_LATCH1 1622 signal.

25 Fig.2E schematically represents a second level latch P1_LATCH 1640. Fig.2F schematically represents a second level latch P2_LATCH 1650.

30 Both second level latches P1_LATCH 1640 and P2_LATCH 1650 receive as input the CLK 1512 signal, and the A_LATCH1 1612 signal and the B_LATCH1 1622 signal from respective first level latches IOB/LATCH1 1610 and IOB/LATCH2 1620 represented in Figs. 2B and 2C.

Second level latch P1_LATCH 1640 generates, as output, signals AP1_LATCH 1642 and BP1_LATCH 1644. Second level latch P2_LATCH 1650 generates, as output, signals AP2_LATCH 1652 and BP2_LATCH 1654.

Second level latch P1_LATCH 1640 latches the data using the rising edge of the CLK 1512 signal, while P2_LATCH 1650 latches the data using the falling edge of the CLK 1512 signal.

5

The 10-bit control signals and the 64-bit data signal from the OC192 INTERFACE 2000 are synchronously triggered, for both the first level latch IOB/LATCH1 1610, IOB/LATCH2 1620 and the second level latch P1_LATCH 1640, P2_LATCH 1650.

10 While the data being triggered from the second level latch P1_LATCH 1640, P2_LATCH 1650, the transmitting frequency in the TRANSFORMER 1000 is half of the transmitting frequency in the OC192 INTERFACE 2000.

Fig. 2D schematically represents a generic FIFO-X DATA FLOW CONTROLLER 1410 that sends the internal FIFO status (FIFO_STATUS 1412) and external FIFO status (TX_CLAV/PA 3010) to the OC192 INTERFACE 2000. Once the internal FIFO or the external FIFO is full or nearly full (predefined by connecting physical layer chips), a predefined number of data values are received.

20 In Fig. 2D, FIFO-X DATA FLOW CONTROLLER 1410 generically represents the data flow controller for FIFO1 1660 and FIFO2 1670. FIFO1 1660 and FIFO2 1670 are synchronous FIFOs. The frequency of their WR/RD signals is in the range of 77 MHz to 100 MHz.

25 Figs. 3A to 3F are a second subset of schematic representations of respective parts of the TRANSFORMER 1000.

Fig. 3A schematically represents a FIFO1/FIFO2 WRITE CONTROLLER 1700 that generates respective write signals FIFO1_WR 1702 and FIFO2_WR 1704 for FIFO1 1660 and FIFO2 1670. The FIFO1/FIFO2 WRITE CONTROLLER 1700 receives as input the CLK 1512 signal, and the AP1_LATCH 1642 and AP2_LATCH 1652 signals from the second level latches represented in Figs. 2E and 2F. The FIFO1/FIFO2 WRITE CONTROLLER 1700 also receives as input FIFO1_STATUS 1662 and FIFO2_STATUS 1672 signals that are generated as described below.

30

Figs. 3B and 3C respectively represent FIFO1 1660 and FIFO2 1670. In Fig. 3B, FIFO1 1660 receives as input the following signals: **FIFO1_WR** 1702, **AP1_LATCH** 1642 and **BP1_LATCH** 1644. In Fig. 3C, FIFO2 1670 receives as input the following signals: **FIFO2_WR** 1704, **AP2_LATCH** 1652 and **BP2_LATCH** 1654. In both cases, the **FIFO1_WR** 1702 and **FIFO2_WR** 1704 signals represent control signals from the FIFO1/2 WRITE CONTROLLER 1700 of Fig. 3A. **AP1_LATCH** 1642 and **BP1_LATCH** 1644 signals, and **AP2_LATCH** 1652 and **BP2_LATCH** 1654 signals represent fully latched input data values.

FIFO1 1660 and FIFO2 1670 generate respective output control signals **FIFO1_STATUS** 1662 and **FIFO2_STATUS** 1672, both of which are controlled by respective input FIFO read control signals **FIFO1_RD** 1712 and **FIFO2_RD** 1714. The output of the FIFOs 1660 and 1670 is **FIFO1_AP** 1664 and **FIFO1_BP** 1666 for FIFO1 1660 and **FIFO2_AP** 1674 and **FIFO2_BP** 1676 for FIFO2 1670.

With reference to Figs. 3A to 3C, the incoming serial data stream on the OC192 INTERFACE 2000 is divided into two parallel data paths for respective FIFOs 1660 and 1670. Accordingly, the depth of the respective FIFOs 1660 and 1670 may not be equal and respective data locations in the two different FIFOs 1660 and 1670 may not be in alignment. Consequently, when data is read out from the FIFOs 1660 and 1670, the data may not be in appropriate order. This phenomenon is referred to as data "zigzag" in the space domain and the time domain. Data zigzag is described in further detail in the subsection entitled "Data inserting mechanism", with reference to Fig. 7.

When the FIFO write signals **FIFO1_WR** 1702 and **FIFO2_WR** 1704 are generated, a data inserting mechanism is implemented in the FIFO1/FIFO2 WRITE CONTROLLER 1700. As the respective depths of the data feeding into the two serial data paths are not equal, extra non-informational (that is, dummy) data values are expediently written into the FIFO (1660 or 1670) that has a shallower data depth, to compensate for the mismatch in respective data depths. The process by which this occurs is described in further detail in the subsection entitled "Data inserting mechanism"

FIFO1 1660 and FIFO2 1670 are implemented in synchronous mode, as described in the subsection entitled "FIFO data structures". With the chosen data structures, implementation of control logic is simplified. As a result, the FIFOs 1660 and 1670 can operate at higher working frequencies. A practical working frequency is in the range 77
5 MHz to 100 MHz, which is half the transmitting frequency on the OC192 INTERFACE 2000.

Fig. 3E schematically represents a FIFO1/FIFO2 READ CONTROLLER 1710. The FIFO1/FIFO2 READ CONTROLLER 1710 receives as input the clock signal **CLK** 1512,
10 and other control signals **FIFO1_STATUS** 1662 and **FIFO2_STATUS** 1672. The FIFO1/FIFO2 READ CONTROLLER 1710 generates as output: **FIFO1_RD** 1712, and **FIFO2_RD** 1714. Control signals **AB_CLK** 1716, **RD_VALID** 1717, and **LATCH_CLK** 1718 are also generated.

15 Fig. 3D schematically represents a NEGOTIATOR 1720 that receives as input the following control signals: **AB_CLK** 1716, **RD_VALID** 1717, and **LATCH_CLK** 1718. The NEGOTIATOR 1720 also receives as input the following data signals: **FIFO1_AP** 1664, **FIFO1_BP** 1666, **FIFO2_AP** 1674 and **FIFO2_BP** 1676. The NEGOTIATOR 1720 generates as output signals **N_AP** 1722 and **N_BP** 1724.

20 Fig. 3F schematically represents the interconnection to the OC48 INTERFACE 3000. A clock signal **TX_CLK** 3020 is supplied to the OC48 INTERFACE 3000, and data is passed to the OC48 INTERFACE 3000 via **TX_ENB** 3030. Control signal **TX_CLAV/PA** is sent from the OC48 INTERFACE 3000 to the TRANSFORMER 1000.

25 Though FIFO1 1660 and FIFO2 1670 are synchronous FIFOs, their writing and reading taps are different. While data is read from the FIFOs, the NEGOTIATOR 1720 is implemented for triggering the read out of all data simultaneously, and deciding the order in which data is transmitted from the FIFOs 1660 and 1670.

30 For the simultaneously triggered data in NEGOTIATOR 1720, data is transmitted in same order of the incoming data stream from the OC192 INTERFACE 2000. As a consequence, the negotiating signal **AB_CLK** 1716 is implemented in FIFO1/FIFO2 READ CONTROLLER 1710.

AB_CLK 1716 active low controls the data output from **FIFO1** to the **TAP GENERATOR 1800** and **AB_CLK 1716** active high controls the data output from **FIFO2** to the **TAP GENERATOR 1800**.

5

The transmission control signals are connected to the **FIFO1/FIFO2 READ CONTROLLER 1710** to generate the **AB_CLK 1716** signal. Signals such as **FIFO1_AP {SOCP, EOP, VALID, FIFO1_CHANNEL [3:0]}**, **FIFO2_AP {SOCP, EOP, VALID, FIFO2_CHANNEL [3:0]}** are used to determine the order in which data is transmitted.

10

Data is first transmitted from **FIFO2_AP 1674**, **FIFO2_BP 1676**, if **AB_CLK 1716** is active high. Data is first transmitted from **FIFO1_AP 1664**, **FIFO1_BP 1666**, if **AB_CLK 1716** is active low. This selection of data for transmission is described in further detail in the subsection entitled "Negotiation mechanism".

15

The **RD_VALID 1717** signal provides further information as to whether the input data fed into **NEGOTIATOR 1720** is valid data. When **RD_VALID 1717** is active high, data output from the **FIFO** with each **FIFO** read operation can be regarded as valid data values and transmitted, otherwise the outputs of the **NEGOTIATOR 1720** are all zero. Also, the
20 extra insert data values can be filtered out once the input **VALID** signal and **CHANNEL [3:0]** signals are received. Refer to the subsection entitled "Removing additional data".

Fig. 4 is a third subset of schematic representations of the **TRANSFORMER 1000**. The
25 input of the **TAP GENERATOR 1800** is **N_AP 1722**, which is a 10-bit control signal originating from negotiator 1720. **N_BP 1724** represents 64-bit data signals originating from **NEGOTIATOR 1720**. The clock **CLK 1512** signal is also supplied to the **TAP GENERATOR 1800**. The **TAP GENERATOR 1800** transforms the 64-bit data to 32-bit data and generates associated control signals on the **OC48 INTERFACE 3000**. If the
30 validity of **N_AP_SIZE [2:0]** is for 64 bits, then the MSB 32-bit data is transmitted first, and the LSB 32-bit data is transmitted later.

Otherwise, the second negotiation mechanism is implemented, if a determination is made that only a portion of the 64-bit data is valid. In this case, the valid portion is transmitted

first when the data transmission is finished. Refer to the subsection entitled "Tap controller" for further details.

For the parallel FIFO structure, FIFO1 1660 and FIFO2 1670 are implemented in
5 synchronous mode. Before sending data out through the OC48 INTERFACE 3000, the
64-bit data stream from FIFO1 1660 or FIFO2 1670 is transformed into a 32-bit data
stream, depending on the status of the STATE MACHINE 1810 in Fig. 12. The 64-bit
data stream originates from the OC192 INTERFACE 2000.

10 For data values at the start of the transmitting packet, or data values in the middle of
transmission, if all the 64-bit data values are valid, the MSB 32 bits are sent first to the
OC48 INTERFACE 3000, and the LSB 32 bits follow.

For the last valid data values at the end of the transmitting packet, another negotiation
15 mechanism is presented for determining which part of the 64-bit data values should be
transmitted and when is the end of the packet occurs.

Data structure of FIFOs

20 Control signals and data signals on the OC192 INTERFACE 2000 are triggered into
FIFOs of the TRANSFORMER 1000. The depth and width of these FIFOs is specified as
follows.

All the 64-bit data signals and 16-bit control signals on the OC192 INTERFACE 2000 are
25 passed into these FIFOs. Accordingly, the width is $64 + 16 = 80$ bits. The 16-bit control
signal originating from the OC192 INTERFACE 2000 comprises: **SRX_EOP** 2080,
SRX_ERROR 2090, **SRX_PRTY[3:0]** 2060, **SRX_SIZE[2:0]** 2100,
SRX_CHANNEL[3:0] 2030, **SRX_PRTY[4]** 2070, **SRX_VALID** 2020.

30 The 16-bit control signals are synchronously passed to FIFO1 1660 and FIFO2 1670 with
the data **SRX_DATA_OUT[63:0]** 2050. The purpose of such structure is for the
convenience of further tap controlling on the OC48 INTERFACE 3000.

For example, the last valid data comprises both the MSB 32-bit data and a portion of the LSB 32-bit data. For the OC192 INTERFACE 2000, **SRX_VBYTE**, for example, **SRX_DATA[63:0]** 2050, **SRX_EOP** 2080, **SRX_ERROR** 2090 signals appear at the same time. However, the data and control signals should appear at a different tap for the control signals on the OC48 INTERFACE 3000. The last valid data is sent at one tap (for example, **SRX_SIZE[2:0]** 2100 outputs a sequence 001, 010, 011, 100) or is sent at two continuous taps (for example, **SRX_SIZE[2:0]** 2100 outputs a sequence 000, 101, 110, 111).

Fig. 5 represents a data structure of either FIFO 1660 or 1670, depicted generically as FIFO-X for X = 1 or 2. Signal A 1210 is used to represent the triggered 16-bit control signals on the OC192 INTERFACE 2000, while signal B 1220 is used to represent the triggered 64-bit data value. The FIFO 1660 or 1670 receives as input the following signals:

15

- **SRX_SOC/P** 2040 , a start of cell/packet signal

- **SRX_EOP** 2080, a end of packet signal

20

- **SRX_VALID** 2020 , valid signal for data.

- **SRX_CHANNEL[3:0]** 2030, signal which carries the channel information

- **SRX_ERROR** 2090, an error control signal

25

- **SRX_PRTY[3:0]** 2060, a 4-bit wide parity signal for data

- **SRX_PRTY[4]** 2060, special parity signal for controls

30

- **SRX_SIZE[2:0]** 2100, a 3-bit wide signal that gives the information of valid part of data

- **SRX_DATA_OUT[63:0]** 2050, 64-bit wide incoming data signal

- **FIFOX_WR:** **FIFO1_WR** 1702, **FIFO2_WR** 1704, are generated from **WRITE CONTROLLER** 1700
- **FIFOX_STATUS:** **FIFO1_STATUS** 1662, **FIFO2_STATUS** 1672 are generated from **FIFO1** 1660 and **FIFO2** 1670
- **FIFOX_RD:** **FIFO1_RD** 1712, **FIFO2_RD** 1714 are generated from **READ CONTROLLER** 1710

10 In Fig. 5, the **APX_LATCH** 2021 signals represented in Fig. 5 derive from the 16-bit control signals on **OC192 INTERFACE** 2000, and the **BPX_LATCH** 2051 signals derive from the 64-bit data values on the **OC192 INTERFACE** 2000.

The **FIFO** 1660 or 1670 generates as output signals **A** 1210 and **B** 1220.

15

Figs. 6A and 6B are corresponding timing diagrams that both represents samples of the function of the **TAP CONTROLLER** described directly above.

Fig. 6A represents a case in which valid data values of the last packet on **OC192 INTERFACE** 2000 are on the **MSB** part. So, while data values are transformed into 32-bit data values on **OC48 INTERFACE** 3000, the **VALID[63:48]** values on **OC192 interface** 2000 send to **VALID[31:16]** on **OC48 INTERFACE** 3000 directly.

Fig. 6B represents a case in which valid data values of the last packet on **OC192 INTERFACE** 2000 are on both the **MSB** part and **LSB** part. So, while data values are transformed into 32-bit data values **OC48 INTERFACE** 3000, the **VALID[63:16]** signal on **OC192 INTERFACE** 2000 needs to be transmitted twice. That is, **VALID[63:32]** on **OC192 INTERFACE** 2000 is transmitted to **VALID[31:0]** on **OC48 INTERFACE** 3000 in first (**TAP1**), then the **VALID[31:16]** on **OC192 INTERFACE** 2000 is transmitted to **VALID[31:16]** on **OC48 INTERFACE** 3000 in second (**TAP2**).

Data inserting mechanism

The data stream triggered from OC192 INTERFACE 2000 is divided into two parallel data streams. The number of valid data values that feed data into the different data paths may not be equal.

5 Fig. 7 is a timing diagram that depicts a case of data "zigzag". In Fig. 7, note that: (i) **AP1_VALID** 1644, **AP1_SOC/P** 1649, **AP1_EOP** 1646, **BP1_DATA** 1645 are passed to FIFO1 1660, and (ii) **AP2_VALID** 1654, **AP2_SOC/P** 1659, **AP2_EOP** 1656, **BP2_DATA** 1655 are passed to FIFO2 1670.

10 **SRX_CLK** 2010 is the transmitting clock signal on the OC192 INTERFACE 2000. Data values are valid on the rising edge of the **SRX_CLK** 2010 signal.

SRX_VALID 2020 is the transmitting data valid signal on the OC192 INTERFACE 2000. **SRX_SOCP** 2040 represents the start of packet in transmission. **SRX_EOP** 2080
15 represents the end of packet in transmission. **SRX_DATA_OUT** [63:0] 2050 is a 64-bit data bus on the OC192 INTERFACE 2000.

A1_VALID 1232, **A1_SOCP** 1234, **A1_EOP** 1236 are signals triggered by first level latch IOB/LATCH1 1610. These signals are the output of control signal bus, **A_LATCH1**
20 1612.

B1_DATA 1623 are the 64 bits triggered data, **B_LATCH1** 1622, which are sent from first level latch IOB/LATCH2 1620.

25 **AP1_VALID** 1644, **AP1_SOC/P** 1649, **AP1_EOP** 1646, **BP1_DATA** 1645 are data triggered from **P1_LATCH** 1640. The data are triggered using the rising edge of the **CLK** 1512 signal. **AP2_VALID** 1654, **AP2_SOCP** 1659, **AP2_EOP** 1656, **BP2_DATA** 1655 are data signals triggered from **P2_LATCH** 1650. These data signals are triggered using the falling edge of **CLK** 1512.

30

In Fig. 7, the incoming data values 1, 2, 3, 4, 5 and A, B, C, D, E are divided into two different groups. Data values 1, 3, 5, A, C, E is sent to FIFO1 1660. Data values 2, 4, B, D are sent to FIFO2 1670. In this way, the depth of the FIFO1 1660 is 6, whereas, the depth of FIFO2 1670 is 4. Accordingly, two extra dummy data values are inserted into

FIFO2 1670 owing to this difference in data depths between the two FIFOs 1660 and 1670.

The data inserting mechanism is implemented in FIFO1/FIFO2 WRITE CONTROLLER 1700. The FIFO write signals are generated based on the FIFO STATUS, and the triggered control signals. The following control signals are used to generating the FIFO write signal:

- **AP1_VALID** 1644
- **AP1_EOP** 1646
- **AP1_CHANNEL [3:0]** 1648
- **AP2_VALID** 1654
- **AP2_EOP** 1656
- **AP2_CHANNEL [3:0]** 1658

Fig. 8 schematically represents the interface and internal functional blocks of FIFO1/FIFO2 WRITE CONTROLLER 1700.

EVEN COUNTER1 1730 counts the number of data values written into the FIFO1 1660 in even or odd mode. If signal **A_EVEN** 1734 is low, then an even number of data values are written into FIFO1 1660. If **A_EVEN** 1734 is high, then an odd number of data is written into FIFO1 1660.

EVEN COUNTER 2 1750 counts the number of data values written into the FIFO2 1670 in even or odd mode. If signal **B_EVEN** 1754 is low, then an even number of data values is written into FIFO2 1670. If **B_EVEN** 1754 is high, then an odd number of data values are written into FIFO2 1670. **A_EVEN_TRIG** 1732 is triggered by **A_EVEN** 1734 using the falling edge of **CLK** 1512. **B_EVEN_TRIG** 1752 is triggered by **B_EVEN** 1754 using the rising edge of **CLK** 1512.

With the input signals **AP1_CHANNEL[3:0]** 1648, **CHANNEL-DECODER1** 1740 acts as a 4-bit decoder to generate the **CHANNEL1** 1742 signal. With the input signals **AP2_CHANNEL[3:0]** 1658, **CHANNEL-DECODER2** 1760 acts as a 4-bit decoder to generate the **CHANNEL2** 1762 signal. **CHANNEL-DECODER1** 1740 and **CHANNEL-DECODER2** 1760 have the same decoding function.

EOP_TRIGGER 1770 triggers the **AP1_EOP** 1646 and **AP2_EOP** 1656 signals using the falling edge and rising edge of **CLK** 1512 independently.

LOGIC1 1780 and **LOGIC2** 1790 are logic circuits that (independently) implement the following functions for **FIFO1_WR** 1782 and **FIFO2_WR** 1792 respectively.

$$\begin{aligned} \text{FIFO1_WR} = & \text{AP1_VALID AND} \\ & \text{CHANNEL1 OR} \\ & \text{AP2_EOP_TRIG AND} \\ & (\text{B_EVEN XOR A_EVEN_TRIG}) \quad (1) \end{aligned}$$

$$\begin{aligned} \text{FIFO2_WR} = & \text{AP2_VALID AND} \\ & \text{CHANNEL2 OR} \\ & \text{AP1_EOP_TRIG AND} \\ & (\text{A_EVEN XOR B_EVEN_TRIG}) \quad (2) \end{aligned}$$

When the respective data depths of the two FIFOs are not equal, then the write signal of the FIFO that has less data depth extends (that is, remains high) for another one write cycle. Equations (1) and (2) provide this function. Consequently, an extra data value is fed into the respective FIFO by this high write signal. This mechanism maintains the order of the original data stream.

Fig. 9 is a timing diagram for the signals discussed above, involving the **FIFO1_WR** 1782 and **FIFO2_WR** 1792 write signals of **FIFO1** 1660 and **FIFO2** 1670 respectively.

Negotiation mechanism

Fig. 10 schematically represents the structure of the FIFO1/FIFO2 READ CONTROLLER 1710 shown in Fig. 3E.

- 5 The first negotiation mechanism is implemented in FIFO1/FIFO2 READ CONTROLLER 1710, and provides data realignment for the OC192 data stream. The negotiation mechanism generates the selection signal **AB_CLK** 1716. If **AB_CLK** 1716 is low, then the data values read out from FIFO1 1660 are transmitted to the output of NEGOTIATOR 1720. If **AB_CLK** 1716 is high, then the data values read out from FIFO2 1670 are
10 transmitted to the output of NEGOTIATOR 1720.

Fig. 10 schematically represents the interface of FIFO1/FIFO2 READ CONTROLLER 1710 and its internal function blocks.

- 15 **CLK33 GENERATOR** 1430 is a clock divider. The frequency of the outgoing clock signal **CLK33** 1432 is half the frequency of the incoming clock signal **CLK** 1512. **CHANNEL-DECODER1** 1410 and **CHANNEL-DECODER2** 1420 decode the input signals **FIFO1_CHANNEL**[3:0] 1668 and **FIFO2_CHANNEL**[3:0] 1678.

- 20 **RD_GENERATOR** 1440 generates read signals for FIFO1 1660 and FIFO2 1670. If the FIFOs 1660 and 1670 are not empty and **TX_CLAV/PA** 3010 is valid (that is, the OC48 interface can receive data) then the read signals are sent to FIFOs 1660 and 1670.

- CLRA GENERATOR** 1460 and **RSTB GENERATOR** 1470 generate the clear and reset
25 signals respectively for signal **CLK33** 1432. **CLRA** 1462 and **RSTB** 1472 are triggered using the falling edge of **CLK33** 1432 to generate **CLRA_TAP** 1463 and **RSTB_TAP** 1473 signals. If **CLRA_TAP** 1463 is valid, then **AB_CLK** 1716 is set to low and if **RSTB_TAP** 1473 is valid, then **AB_CLK** 1716 is set to high. The clear and reset operations are implemented in asynchronous mode. **RSTB_TAP** 1473 and **CLRA_TAP**
30 1463 are internal signals of **SEL_TRIGGER** 1450.

Fig. 11 is a timing diagram involving the signals discussed above. **CLRA** 1462 and **RSTB** 1472 signals are implemented using the following equations:

CLRA = **FIFO1_SOCP AND (NOT FIFO2_EOP) AND**
(NOT FIFO2_SOCP) OR FIFO1_EOP AND
(FIFO2_SOCP OR
(NOT FIFO2_VALID) OR
5 **(NOT FIFO2_CHANNEL))** (3)

RSTB = **FIFO2_SOCP AND**
(NOT FIFO1_EOP) AND
10 **(NOT FIFO1_SOCP) OR**
FIFO2_EOP AND
(FIFO1_SOCP OR
(NOT FIFO1_VALID) OR
(NOT FIFO1_CHANNEL)) (4)

15 Fig. 11 is a timing diagram for FIFO1/FIFO2 READ CONTROLLER 1710. The **CLRA** 1462 signal and **RSTB** 1472 signal are triggered using the **RD_CLK** 1513 signal, to generate the associate reset tap and clear tap for **CLK33** 1432. **RD_CLK** 1513 is generated from **CLK** 1512. Every four cycles of **CLK** 1512 generate one active high
20 pulse. The valid period of this high level is equal to one cycle period of **CLK** 1512.

From Fig. 11, **AB_CLK** 1716 is cleared and reset by the **CLRA_TAP** 1463 and **RSTB_TAP** 1473 signals. If **AB_CLK** 1716 is low, then **FIFO1_TRIG** 1515 data value is transmitted. If **AB_CLK** 1716 is high, then **FIFO2_TRIG** 1516 data value is transmitted.
25 **FIFO1_TRIG** 1515 and **FIFO2_TRIG** 1516 are triggered from **FIFO1_READ_DATA** 1443 and **FIFO2_READ_DATA** 1445 signals respectively, though independent use of **RD_CLK** 1513. **FIFO1_READ_DATA** 1443 are the data read out from **FIFO1** 1660. **FIFO2_READ_DATA** 1445 are the data read out from **FIFO2** 1670. **FIFO1_RD** 1442 is the read enable signal of **FIFO1** 1660. **FIFO2_RD** 1444 is the read enable signal of
30 **FIFO2** 1660. **FIFO1_RD** and **FIFO2_RD** signals are the FIFO read control signals, and **FIFO1_READ_DATA** and **FIFO2_READ_DATA** are the 64-bit data values read out from the respective FIFOs.

Filtering out dummy data values

Rearranged data values from NEGOTIATOR 1720 provide the original order of the 64-bit data stream from the OC192 INTERFACE 2000. However, as extra data values may have been included in the data stream in accordance with the above described data inserting mechanism, any such extra data values are removed or filtered from the data stream. The identification of extra inserted data values is determined as follows.

The signals that come from NEGOTIATOR 1720 include control signals such as **N_AP_SOCP** 1314, **N_AP_EOP** 1316, etc. If the **N_AP_VALID** 1312 is invalid, then this data value is an inserted data value. If the decoded signal of the **N_AP_CHANNEL** [3:0] is invalid, then this data value is determined to be an inserted data value.

The signals that come from NEGOTIATOR 1720 are already filtered. In order to do the filtering, a judgement is made using the control signals that are noted as **FIFO1_AP** 1664, **FIFO2_AP** 1674. If **FIFO1_VALID** 1665 is invalid, then the data **FIFO1_BP** 1666 is invalid. If **FIFO1_VALID** 1665 is valid and **FIFO1_CHANNEL** [3:0] 1668 are both valid, then **FIFO1_BP** 1666 is valid. Otherwise, **FIFO1_BP** 1666 is invalid. This filtering mechanism provided for FIFO1 is the same for FIFO2.

This mechanism is implemented in NEGOTIATOR 1720. Before the negotiator triggers out the **N_AP** and **N_BP** signals, the extra inserted dummy data value is filtered out.

Tap controller

Fig. 12 schematically represents the structure of the TAP GENERATOR 1800 shown in Fig. 4.

The TAP GENERATOR 1800 latches the data and control signals from NEGOTIATOR 1720. And input signals such as: **N_AP_VALID** 1312, **N_AP_SIZE** [2:0] 1318 are sent to the internal STATE MACHINE 1810. Fig. 12 schematically represents the interface and internal functional blocks of TAP GENERATOR 1800.

STATE MACHINE 1810 generates 3 states: **IDLE** 1814, **TAP1** 1816, and **TAP2** 1818. For the output controller **T_LATCH2** 1830, if the status is **IDLE** 1814, then all the output signals except **TX_ENB** 3030 are set to zero. Transmitting valid signal **TX_ENB** 3030 is set to high, which means that the transmitting is invalid or paused. If the status from the state machine 1810 is **TAP1**, then the MSB 32 bit are transmitted. If the status is **TAP2** 1818, then the LSB 32 bit are transmitted.

If the status of the STATE MACHINE 1810 is **TAP1** 1816, the output data is triggered from the input interface of TAP GENERATOR 1800. If the status of the state machine 1820 is **TAP2** 1818, the output data is triggered from the internal latched data, **N_AP_EOP** 1316, **N_AP_SIZE[2:0]** 1318, **N_AP_ERR** 1320, **N_AP_DATA[31:0]** 1332. Control signals of the OC48 INTERFACE 2000, such as **TX_ENB** 3030, **TX_SOCP** 3050, **TX_EOP** 3040, **TX_ERROR** 3060, **TX_VBYTE[2:0]** 3090 are generated in the manner indicated by the psuedo-code of Fig. 13.

Fig. 15 is a state transition diagram for the STATE MACHINE 1810. Two signals, **N_AP_VALID** 1312 and **N_AP_SIZE[2:0]** 1318, are used as stimulus for the state transition in the STATE MACHINE 1810.

Parity generator

Parity signal generator for **T_LATCH2** 1830 shown in Fig. 12 is described as follows. The inputs for parity generator are: p_0, p_1, p_2, p_3 in which p_0, p_1, p_2, p_3 are parity signals of different data segments.

These parity signals are described as follows:

$$p_0 = x_{15} \text{ XOR } x_{14} \text{ XOR } \dots \text{ XOR } x_0 \quad (5)$$

$$p_1 = x_{31} \text{ XOR } x_{30} \text{ XOR } \dots \text{ XOR } x_{16} \quad (6)$$

$$p_2 = x_{47} \text{ XOR } x_{46} \text{ XOR } \dots \text{ XOR } x_{32} \quad (7)$$

$$p_3 = x_{63} \text{ XOR } x_{62} \text{ XOR } \dots \text{ XOR } x_{48} \quad (8)$$

The output for the parity generator is as follows.

$$\text{P-first} = p_3 \text{ XOR } p_2 \quad (9)$$

$$\text{P-second} = p_1 \text{ XOR } p_0 \quad (10)$$

In expression (9) above, the MSB data ($x_{63}, x_{62}, \dots x_{32}$) is transmitted first, while in
5 expression (10) above the LSB data ($x_{31}, x_{30}, \dots x_0$) is transmitted second.

Digital hardware implementation

The described techniques are implemented in digital logic in the manner described above.
10 The hardware is incorporated on a single FPGA chip that can be fabricated on a single ASIC
using high speed CMOS.

Both FIFOs 1660 and 1670 in the above described implementation use the primitive
BRAM4_S16_S16 in VIRTEX-E (Part Number : XCV100E -8 560 C, Manufacturer :
15 XILINX, Inc).

The GLOBAL CLOCK BUFFER 1500 is a primitive in a XILINX FPGA chip. IOB
1630, 1910, 1920, refers the primitives of input/output buffers in XILINX FPGA (XCV1
OOOE -8 BG560 C).

20 All the blocks are implemented using the XST VHDL compiler in XILINX ISE 3.3.08i.
And the location of the internal circuits are allocated using the Xilinx FloorPlanner,
which is provided within the ISE software.

Design structure

Fig 16 illustrates the overall structure and implementation of the design described above.
the components and signals depicted in Fig. 16 are described above with reference to
Figs. 1 to 15.

30 Data and control signals from the OC192 INTERFACE 2000 are fed into the first level of
latch, which can be merged with IOBs. The GLOBAL CLOCK BUFFER 1500 is used to
buffering the clock and the clock divider, which is implemented by the VHDL coding is
used to get the half of the transmitting frequency. The second level latches input data and

control signals of OC192 INTERFACE 2000 into FIFOs 1660, 1670. FIFO READ CONTROLLER 1710 and FIFO WRITE CONTROLLER 1700 are used to read/write data from/to FIFOs. The NEGOTIATOR 1720 is managing the tasks of filtering the dummy data and forming the data as in the original order of data values from OC192
5 INTERFACE 2000 from the "zigzag" pattern. TAP GENERATOR 1800, which converts the 64-bit data values to 32-bits data values with respect to the control signals and the data can be placed on the OC48 INTERFACE 3000 in one or two taps.

Overview of techniques

10 In overview, the techniques described herein operate as follows with reference to Fig. 17.

An incoming OC 192 data stream is received in step 4010. In step 4020, this incoming data stream is divided into two parallel data streams. Dummy data values are inserted, as
15 appropriate, into the parallel data streams in step 4030. This selective insertion of dummy data values is performed to address data misalignment in the parallel data streams.

In step 4040, the parallel data streams are collated into a single data stream. The collated data stream is filtered to remove the previously inserted dummy data values in step 4050.

20 In step 4060, the filtered data stream is transmitted as an outgoing OC48 data stream.

VHDL implementation

Fig. 18 schematically represents a generic architecture for a suitable network bridge. In
25 contrast with the above description, Fig. 18 illustrates an implementation of the described techniques in software rather than hardware/firmware. Such implementations can be notionally provided by the use of digital logic that is equivalent to a corresponding hardware implementation. In practice, such software/firmware implementations are typically impractical due to speed advantages achieved through hardware implementations
30 of the type described herein.

With reference to Fig. 18, the bridge 5000 has an input interface 5010, an output interface 5060, digital logic 5020, a processor 5030, and associated registers 5040 and memory 5050. The input interface 5010 interfaces to the digital logic 5020, which in turn

interfaces to the output interface 5060. The incoming data stream at the input interface 5010 is processed according to the digital logic 5020 that is executed by the processor 5030, which is connected to the digital logic 5020.

5 The processor 5030, to which storage registers 5040 and a memory 5050 are operatively connected, executes the digital logic 5020 that is essentially a control program stored in the memory 5050. The registers 5040 store values obtained from the processor 5030, during computation by the processor 5030. The processor 5030 executes the digital logic 5020 for the ultimate purpose of data transforming an incoming data stream on the input
10 interface 5010 to an outgoing data stream on the output interface 5060.

The digital logic 5020 and processor 5030 maintain parallel data streams that are processed in a manner analogous to that described in relation to the above hardware implementation.

15

Conclusion

A method, digital hardware and digital logic is described herein for transforming data from a wide data stream to a narrow data stream by use of a network bridge that
20 implements the described techniques.

Though a particular implementation has been described in the context of transforming data from OC192 to OC48 data streams, the described techniques are more broadly applicable to any wider data stream to narrower data stream. The design can be used for
25 addressing data zigzag problems when a parallel structure is used in data transformation.

The problem of data zigzag is recognised, and addressed from the viewpoint of synchronously transmitting data streams. Conventionally, parallel structure assumes that source-synchronous signaling is used to drive clock and data signals from a single device
30 and forward the clock along with the data to the destination interface. Thus, clock and data signals propagate along adjacent paths with matched propagation delays. However, as described herein, the clock and data signals can propagate along adjacent paths with different propagation delays. A parallel structure is used in multi-channel 622 Mb/s

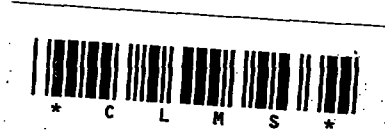
LVDS data transfers for VirtexE Devices and the four parallel-in/serial-out shift registers feeding two double-data-rate registers.

5 While implementing the transformation from 10-Gb/s to 2.5-Gb/s in a FPGA or ASIC, the internal delay of the design is a critical issue. Further, the internal delay leads to zero time margin or negative margin, so clock skew occurs that affects data quality. With the described parallel FIFO structure, internal delay is not a critical problem for floor planning in a FPGA or ASIC. Using the described parallel structure, the internal clock speed is half that of the OC192 INTERFACE 2000. Consequently, a cheaper and slower
10 FPGA or ASCI can be used for data transformation.

The described TRANSFORMER 1000 provides a direct link between a Ganges 10-Gb/s framer, a Yukon, 2.5-Gb/s framer, and a Forte wire-speed edge router IC. Data transformation is a typical application in such router, switcher designs.

15

Various alterations and modifications can be made to the techniques and arrangements described herein, as would be apparent to one skilled in the relevant art.



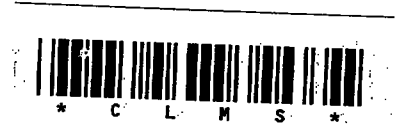
We Claim:

1. A method for transforming an incoming data stream to an outgoing data stream, the bandwidth of the incoming data stream being greater than the bandwidth of the outgoing data stream, the method comprising the steps of:
 - receiving said incoming data stream of regular data values from an input data interface;
 - dividing said incoming data stream into at least two parallel data streams;
 - selectively inserting dummy data values into at least one of said parallel data streams;
 - collating said parallel data streams into a collated data stream;
 - filtering said collated data stream by removing said inserted dummy data values from said collated data stream; and
 - transmitting said filtered data stream as said outgoing data stream of regular data values;

wherein said selective insertion of said dummy data values maintains the order of the regular data values in said incoming and outgoing data streams by avoiding non-alignment of regular data values in said at least two parallel data streams.
2. The method as claimed in claim 1, wherein said at least two parallel data streams are received by respective first-in, first-out (FIFO) hardware components.
3. The method as claimed in claim 2, wherein said respective FIFO hardware components operate in a synchronous data mode.



4. The method as claimed in claim 2, wherein each of said respective FIFO hardware components have the same data capacity.
5. The method as claimed in claim 1, further comprising the step of: generating parity data for error control of regular data values in said outgoing data stream.
6. The method as claimed in claim 5, wherein said parity data is generated using an even parity-generating algorithm.
7. The method as claimed in claim 2, further comprising the step of: latching a read signal from said respective FIFO hardware components during said step of filtering said collated data stream, to prevent dummy data values being treated as regular data values while one of said respective FIFO hardware components is in a pause mode.
8. The method as claimed in claim 1, wherein said regular data values comprise data signals and control signals in said incoming data streams and said outgoing data streams.
9. The method as claimed in claim 1, wherein said incoming data stream is a OC192 data stream and said outgoing data stream is a OC48 data stream.
10. The method as claimed in claim 9, wherein said OC192 data stream is supported by Flexbus4/Pos4 and said OC48 data stream is supported by Flexbus3/Pos3.
11. Digital hardware for transforming an incoming data stream to an outgoing data stream, the bandwidth of the incoming data stream being greater than the bandwidth of the outgoing data stream, the digital hardware comprising:
 - means for receiving said incoming data stream of regular data values from an input data interface;
 - means for dividing said incoming data stream into at least two parallel data streams;



means for selectively inserting dummy data values into at least one of said parallel data streams;

5 means for collating said parallel data streams into a collated data stream;

means for filtering said collated data stream by removing said inserted dummy data values from said collated data stream; and

10

means for transmitting said filtered data stream as said outgoing data stream of regular data values;

15 wherein said selective insertion of said dummy data values maintains the order of the regular data values in said incoming and outgoing data streams by avoiding non-alignment of regular data values in said at least two parallel data streams.

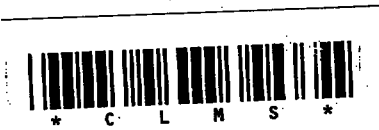
12. 20 Digital logic, embodied in a digital medium and able to be executed by digital hardware able to interpret the digital logic, for transforming an incoming data stream to an outgoing data stream, the bandwidth of the incoming data stream being greater than the bandwidth of the outgoing data stream, the digital hardware comprising:

25 logic means for receiving said incoming data stream of regular data values from an input data interface;

logic means for dividing said incoming data stream into at least two parallel data streams;

30

logic means for selectively inserting dummy data values into at least one of said parallel data streams;



logic means for collating said parallel data streams into a collated data stream;

5 logic means for filtering said collated data stream by removing said inserted dummy data values from said collated data stream; and

logic means for transmitting said filtered data stream as said outgoing data stream of regular data values;

10 wherein said selective insertion of said dummy data values maintains the order of the regular data values in said incoming and outgoing data streams by avoiding non-alignment of regular data values in said at least two parallel data streams.

15 13. Digital logic as claimed in claim 12, wherein said medium comprises a field programmable gate array (FPGA) or application specific integrated circuit (ASIC) device, and said logic means are encoded in memory blocks of the FPGA or ASIC device.

20 14. A network bridge for transforming an incoming, relatively high bandwidth data stream to an outgoing, relatively low bandwidth data stream, the network bridge comprising the digital hardware of claim 11.